

## PRACTICAL-4

CODE:

```
C main.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdbool.h>
4
5  struct Node{
6      int data;
7      struct Node *next;
8  };
9
10 struct Node *head;
11
12
13 void create(int ele){
14     if(head==NULL){
15         head=(struct Node*)malloc(sizeof(struct Node));
16         head->data=ele;
17         head->next=NULL;
18         printf("head is created!\n");
19     }
20     else{
21         printf("Linked list is already Created\n");
22     }
23 }
24 //Insert Functions
25 void insert_end(int ele){
26     struct Node *newnode=(struct Node*)malloc(sizeof(struct Node));
27     struct Node *temp;
28     temp=head;
29     if(head!=NULL){
30         newnode->data=ele;
31         while(temp->next!=NULL){
32             temp=temp->next;
33         }
34         temp->next=newnode;
35         newnode->next=NULL;
36     }
37 }
38
39 }
40
41 void insert_begin(int ele){
42     struct Node *newnode=(struct Node*)malloc(sizeof(struct Node));
43     newnode->data=ele;
44     newnode->next=head;
45     head=newnode;
46 }
47 }
48
49 void insert_between(int ele,int pos){
50     struct Node *newnode=(struct Node*)malloc(sizeof(struct Node));
51     int count=1;
52     struct Node *temp;
53     temp=head;
54     while(temp->next!=NULL && count<pos-1){
55         temp=temp->next;
56         count++;
57     }
58     newnode->next=temp->next;
59     temp->next=newnode;
60     newnode->data=ele;
61 }
```

```

65 int delete_begin(){
66     if (head==NULL)
67     {
68         printf("Linked list not exists\n");
69         return -1;
70     }
71     struct Node *temp=head;
72     int ele=head->data;
73     head=temp->next;
74     free(temp);
75     return ele;
76 }
77 int delete_end(){
78     if (head==NULL)
79     {
80         printf("Linked list not exists\n");
81         return -1;
82     }
83     struct Node *temp=head,*prev;
84     while(temp->next!=NULL){
85         prev=temp;
86         temp=temp->next;
87     }
88     int ele=temp->data;
89     prev->next=NULL;
90     free(temp);
91     return ele;
92 }
93
94 int delete_between(int pos){
95     if (head==NULL)
96     {
97         printf("Linked list not exists\n");
98         return -1;
99     }
100
101     struct Node *temp=head,*prev;
102     int count=1,ele;
103     while(temp!=NULL && count<pos){
104         prev=temp;
105         temp=temp->next;
106         count++;
107     }
108     prev->next=temp->next;
109     free(temp);
110     return ele;
111 }
112
113
114 void traverse(){
115     struct Node *temp;
116     temp=head;
117     while(temp!=NULL){
118         printf("%d -> ",temp->data);
119         temp=temp->next;
120     }
121     printf("NULL\n");
122 }

```

```

124 int input_ele(){
125     int ele;
126     printf("Enter element : ");
127     scanf("%d",&ele);
128     return ele;
129 }
130
131 int main(){
132     int choice;
133     bool again=true;
134     int pos;
135     printf("Enter 1 to create HEAD\n");
136     printf("Enter 2 to insert BEGIN\n");
137     printf("Enter 3 to insert BETWEEN\n");
138     printf("Enter 4 to insert END\n");
139     printf("Enter 5 to delete BEGIN\n");
140     printf("Enter 6 to delete BETWEEN\n");
141     printf("Enter 7 to delete END\n");
142     printf("Enter 8 to Display \n");
143     printf("Enter Any Other key to Stop\n");
144     while (again)
145     {
146         printf("Enter Your Choice : ");
147         scanf("%d",&choice);
148         int elem;
149         switch (choice)
150         {
151             case 1:
152                 create(input_ele());
153                 break;
154             case 2:
155                 insert_begin(input_ele());
156                 break;
157             case 3:
158                 printf("Enter position : ");
159                 scanf("%d",&pos);
160                 insert_between(input_ele(),pos);
161                 break;
162             case 4:
163                 insert_end(input_ele());
164                 break;
165             case 5:
166                 elem=delete_begin();
167                 if(elem!=-1)
168                     printf("%d Element Deleted\n",elem);
169                 break;
170             case 6:
171                 printf("Enter position : ");
172                 scanf("%d",&pos);
173                 elem=delete_between(pos);
174                 if(elem!=-1)
175                     printf("%d Element Deleted\n",elem);
176                 break;
177             case 7:
178                 if(elem!=-1)
179                     printf("%d Element Deleted\n",delete_end());
180                 break;
181             case 8:
182                 traverse();
183                 break;

```

```

184
185             default:
186                 again=false;
187                 break;
188         }
189     }
190 }
191
192     return 0;
193 }

```

Output:

1)Creating Head and append:

```
PS D:\ENGINEERING\DSA_C\PRAC_4> cd "d:\ENG
Enter 1 to create HEAD
Enter 2 to insert BEGIN
Enter 3 to insert BETWEEN
Enter 4 to insert END
Enter 5 to delete BEGIN
Enter 6 to delete BETWEEN
Enter 7 to delete END
Enter 8 to Display
Enter Any Other key to Stop

Enter Your Choice : 1
Enter element : 1
head is created!

Enter Your Choice : 4
Enter element : 2

Enter Your Choice : 4
Enter element : 3

Enter Your Choice : 8
1 -> 2 -> 3 -> NULL
```

2)Insert Begin and in between:

```
Enter Your Choice : 2
Enter element : 0

Enter Your Choice : 8
0 -> 1 -> 2 -> 3 -> NULL

Enter Your Choice : 3
Enter position : 2
Enter element : 8

Enter Your Choice : 8
0 -> 8 -> 1 -> 2 -> 3 -> NULL

Enter Your Choice : █
```

### 3)Delete Begin:

```
ENTER Your Choice : 5
0 Element Deleted

ENTER Your Choice : 8
8 -> 1 -> 2 -> 3 -> NULL
```

### 4)Delete Between:

```
ENTER Your Choice : 8
8 -> 1 -> 2 -> 3 -> NULL

ENTER Your Choice : 6
ENTER position : 3
2 Element Deleted

ENTER Your Choice : 8
8 -> 1 -> 3 -> NULL
```

### 5)Delete End:

```
ENTER Your Choice : 8
8 -> 1 -> 3 -> NULL

ENTER Your Choice : 7
3 Element Deleted

ENTER Your Choice : 8
8 -> 1 -> NULL
```